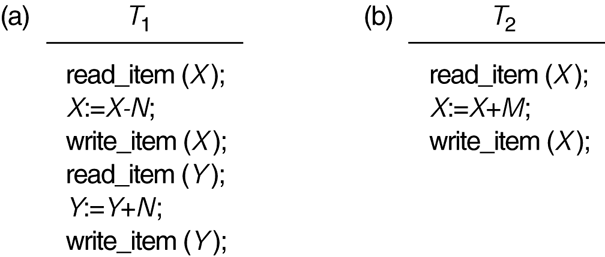
* A **Transaction**:
  + Logical unit of database processing that includes one or more access operations (read -retrieval, write - insert or update, delete).
* A transaction (set of operations) may be stand-alone specified in a high level language like SQL submitted interactively, or may be embedded within a program.
* **Transaction boundaries**:
  + Begin and End transaction.
* An **application program** may contain several transactions separated by the Begin and End transaction boundaries.

SIMPLE MODEL OF A DATABASE (for purposes of discussing transactions):

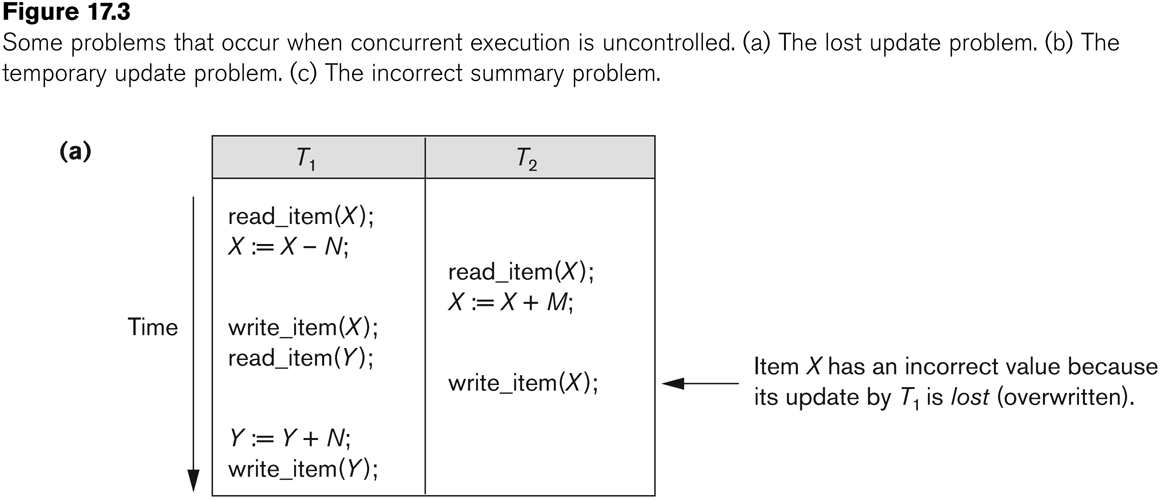
* **A database** is a collection of named data items
* **Granularity** of data - a field, a record , or a whole disk block (Concepts are independent of granularity)
* Basic operations are **read** and **write**
  + **read\_item(X**): Reads a database item named X into a program variable. To simplify our notation, we assume that the program variable is also named X.
  + **write\_item(X**): Writes the value of program variable X into the database item named X.

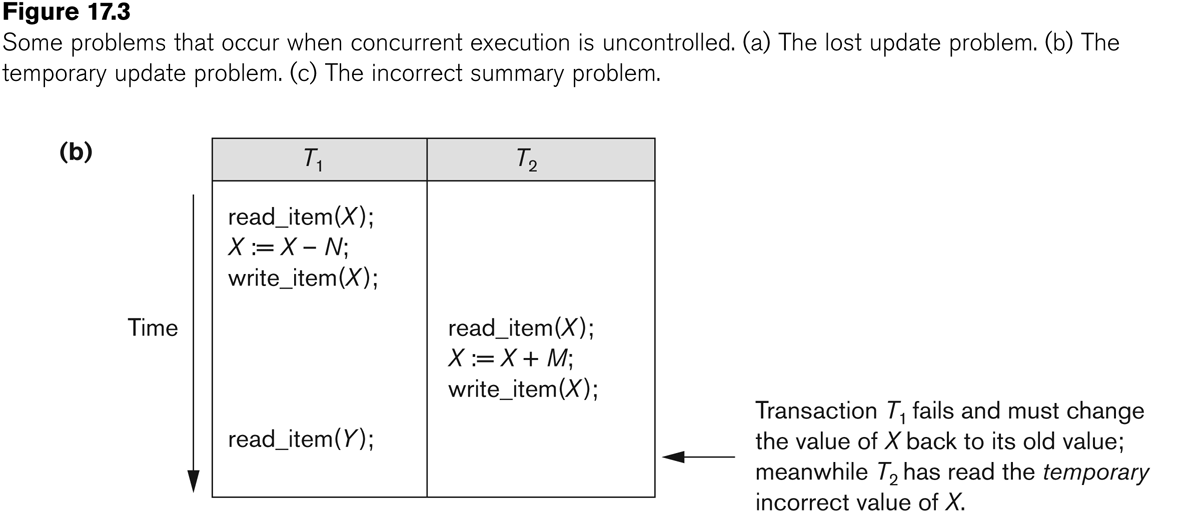
READ AND WRITE OPERATIONS:

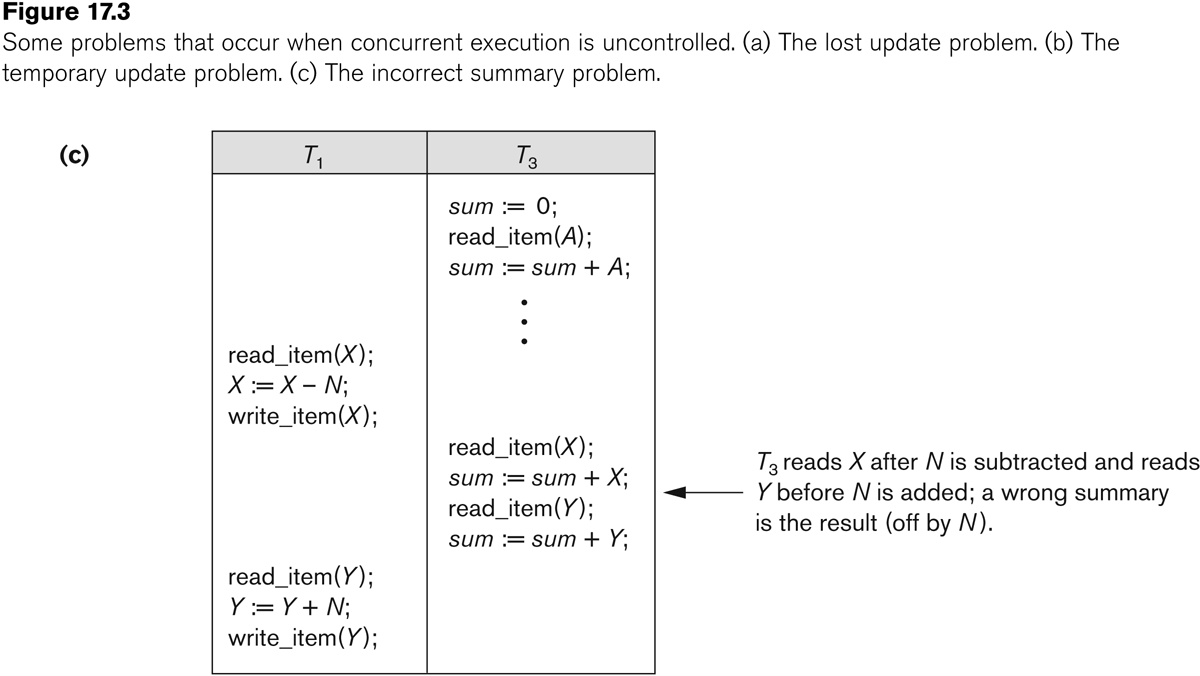
* Basic unit of data transfer from the disk to the computer main memory is one block. In general, a data item (what is read or written) will be the field of some record in the database, although it may be a larger unit such as a record or even a whole block.
* read\_item(X) command includes the following steps:
  + Find the address of the disk block that contains item X.
  + Copy that disk block into a buffer in main memory (if that disk block is not already in some main memory buffer).
  + Copy item X from the buffer to the program variable named X.
* **write\_item(X**) command includes the following steps:
  + Find the address of the disk block that contains item X.
  + Copy that disk block into a buffer in main memory (if that disk block is not already in some main memory buffer).
  + Copy item X from the program variable named X into its correct location in the buffer.
  + Store the updated block from the buffer back to disk (either immediately or at some later point in time).

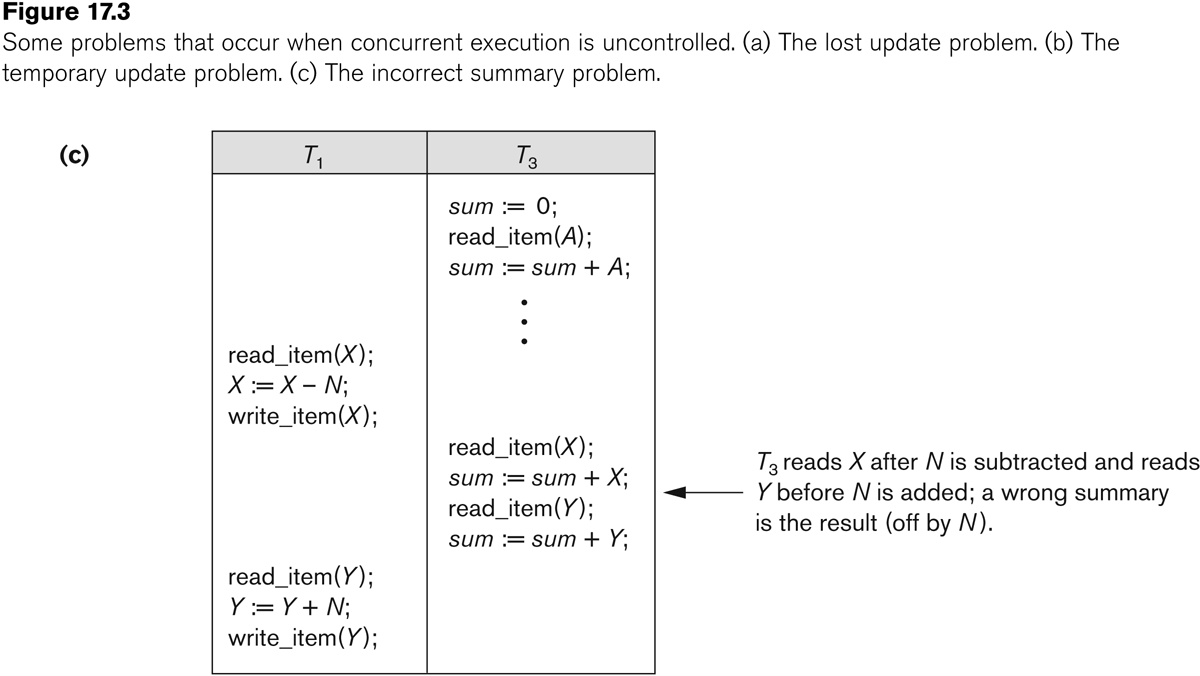


* **The Lost Update Problem**
  + This occurs when two transactions that access the same database items have their operations interleaved in a way that makes the value of some database item incorrect.
* **The Temporary Update (or Dirty Read) Problem** 
  + This occurs when one transaction updates a database item and then the transaction fails for some reason (see Section 17.1.4).
  + The updated item is accessed by another transaction before it is changed back to its original value.
* **The Incorrect Summary Problem**
  + If one transaction is calculating an aggregate summary function on a number of records while other transactions are updating some of these records, the aggregate function may calculate some values before they are updated and others after they are updated.









Why **recovery** is needed:

(What causes a Transaction to fail)

**1. A computer failure (system crash):**

* + - A hardware or software error occurs in the computer system during transaction execution. If the hardware crashes, the contents of the computer’s internal memory may be lost.

**2. A transaction or system error:**

* + - Some operation in the transaction may cause it to fail, such as integer overflow or division by zero. Transaction failure may also occur because of erroneous parameter values or because of a logical programming error.

**3. Local errors or exception conditions detected by the transaction:**

* + - Certain conditions necessitate cancellation of the transaction. For example, data for the transaction may not be found. A condition, such as insufficient account balance in a banking database, may cause a transaction, such as a fund withdrawal from that account, to be canceled. A programmed abort in the transaction causes it to fail.

**4.** **Concurrency control enforcement:**

The concurrency control method may decide to abort the transaction, to be restarted later, because it violates Serializability or because several transactions are in a state of deadlock.

**5. Disk failure:**

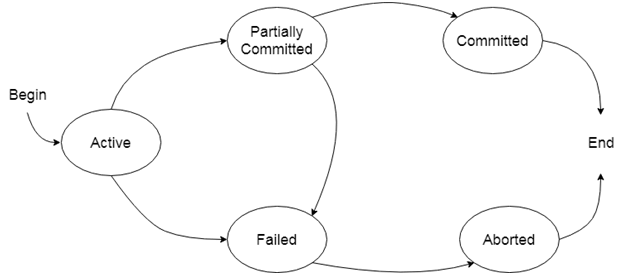
* + - Some disk blocks may lose their data because of a read or write malfunction or because of a disk read/write head crash. This may happen during a read or a write operation of the transaction.

**6. Physical problems and catastrophes:**

* + - This refers to an endless list of problems that includes power or air-conditioning failure, fire, theft, sabotage, overwriting disks or tapes by mistake, and mounting of a wrong tape by the operator.

**States of Transaction**

**In a database, the transaction can be in one of the following states -**



**Active state**

* The active state is the first state of every transaction. In this state, the transaction is being executed.
* For example: Insertion or deletion or updating a record is done here. But all the records are still not saved to the database.

**Partially committed**

* In the partially committed state, a transaction executes its final operation, but the data is still not saved to the database.
* In the total mark calculation example, a final display of the total marks step is executed in this state.

**Committed**

A transaction is said to be in a committed state if it executes all its operations successfully. In this state, all the effects are now permanently saved on the database system.

**Failed state**

* If any of the checks made by the database recovery system fails, then the transaction is said to be in the failed state.
* In the example of total mark calculation, if the database is not able to fire a query to fetch the marks, then the transaction will fail to execute.

**Aborted**

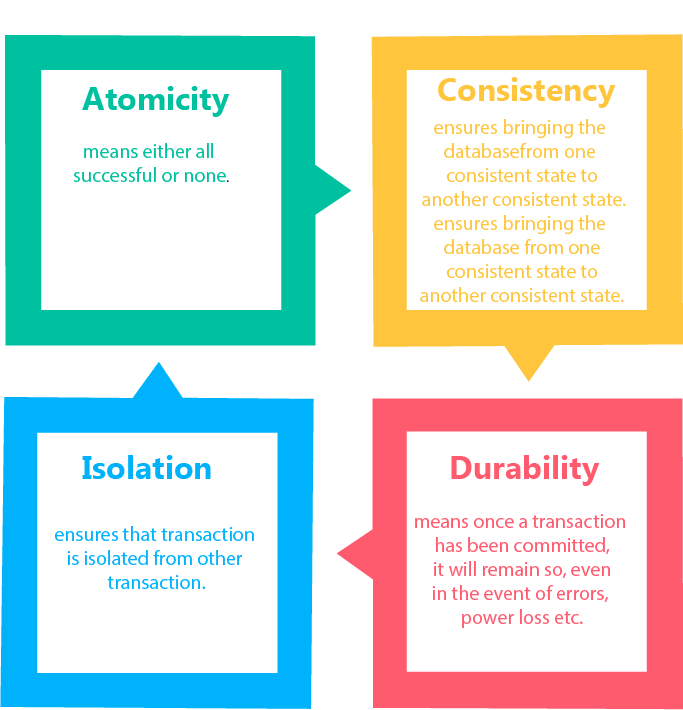
* If any of the checks fail and the transaction has reached a failed state then the database recovery system will make sure that the database is in its previous consistent state. If not then it will abort or roll back the transaction to bring the database into a consistent state.
* If the transaction fails in the middle of the transaction then before executing the transaction, all the executed transactions are rolled back to its consistent state.
* After aborting the transaction, the database recovery module will select one of the two operations:
  1. **Re-start the transaction**
  2. **Kill the transaction**
* **The System Log**
  1. **Log** or **Journal**: The log keeps track of all transaction operations that affect the values of database items.
     + This information may be needed to permit recovery from transaction failures.
     + The log is kept on disk, so it is not affected by any type of failure except for disk or catastrophic failure.
     + In addition, the log is periodically backed up to archival storage (tape) to guard against such catastrophic failures.
  2. T in the following discussion refers to a unique **transaction-id** that is generated automatically by the system and is used to identify each transaction:
  3. **Types of log record:** 
     + **[start\_transaction,T]: Records that transaction T has started execution.**
     + **[write\_item,T,X,old\_value,new\_value]: Records that transaction T has changed the value of database item X from old\_value to new\_value.**
     + **[read\_item,T,X]: Records that transaction T has read the value of database item X.**
     + **[commit,T]:** Records that transaction T has completed successfully, and affirms that its effect can be committed (recorded permanently) to the database.
     + **[abort,T]:** Records that transaction T has been aborted.

# Transaction property

The transaction has the four properties. These are used to maintain consistency in a database, before and after the transaction.

## Property of Transaction

1. **Atomicity**
2. **Consistency**
3. **Isolation**
4. **Durability**



## Atomicity

* It states that all operations of the transaction take place at once if not, the transaction is aborted.
* There is no midway, i.e., the transaction cannot occur partially. Each transaction is treated as one unit and either run to completion or is not executed at all.

Atomicity involves the following two operations:

**Abort:** If a transaction aborts then all the changes made are not visible.

**Commit:** If a transaction commits then all the changes made are visible.

**Example:** Let's assume that following transaction T consisting of T1 and T2. A consists of Rs 600 and B consists of Rs 300. Transfer Rs 100 from account A to account B.

|  |  |
| --- | --- |
| **T1** | **T2** |
| Read(A) A:= A-100 Write(A) | Read(B) Y:= Y+100 Write(B) |

After completion of the transaction, A consists of Rs 500 and B consists of Rs 400.

If the transaction T fails after the completion of transaction T1 but before completion of transaction T2, then the amount will be deducted from A but not added to B. This shows the inconsistent database state. In order to ensure correctness of database state, the transaction must be executed in entirety.

## Consistency

* The integrity constraints are maintained so that the database is consistent before and after the transaction.
* The execution of a transaction will leave a database in either its prior stable state or a new stable state.
* The consistent property of database states that every transaction sees a consistent database instance.
* The transaction is used to transform the database from one consistent state to another consistent state.

**For example:** The total amount must be maintained before or after the transaction.

1. Total before T occurs = 600+300=900
2. Total after T occurs= 500+400=900

Therefore, the database is consistent. In the case when T1 is completed but T2 fails, then inconsistency will occur.

## Isolation

* It shows that the data which is used at the time of execution of a transaction cannot be used by the second transaction until the first one is completed.
* In isolation, if the transaction T1 is being executed and using the data item X, then that data item can't be accessed by any other transaction T2 until the transaction T1 ends.
* The concurrency control subsystem of the DBMS enforced the isolation property.

## Durability

* The durability property is used to indicate the performance of the database's consistent state. It states that the transaction made the permanent changes.
* They cannot be lost by the erroneous operation of a faulty transaction or by the system failure. When a transaction is completed, then the database reaches a state known as the consistent state. That consistent state cannot be lost, even in the event of a system's failure.
* The recovery subsystem of the DBMS has the responsibility of Durability property.